
# An introduction to RooFit

C. Fitzpatrick

For the TPIV students



C. Fitzpatrick

February 16, 2015

# Outline








C. Fitzpatrick



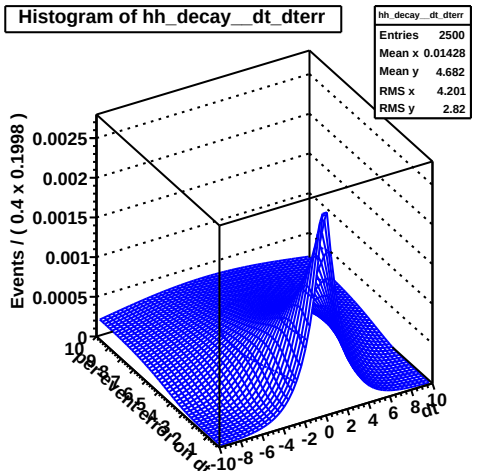
February 16, 2015

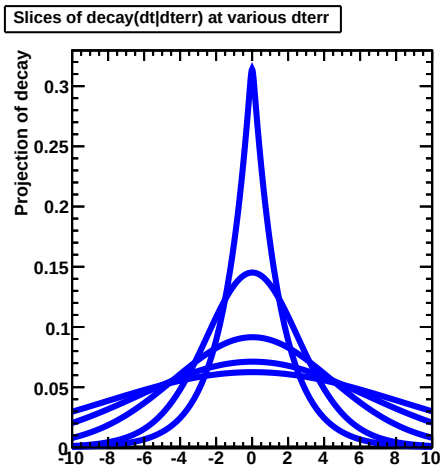


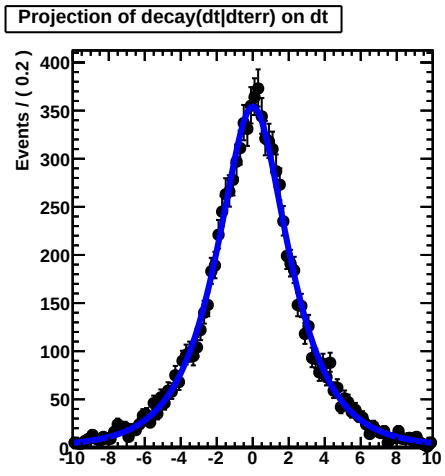
ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

# Introduction

- RooFit is a powerful addition to ROOT as an aid to data modelling and fitting
- RooFit takes a lot of effort out of fitting, but documentation is sparse and outdated
  - I will point you to further documentation and tutorials at the end of the talk
- This talk will introduce the basic concepts to get you started
  - I find examples to be the best way to learn, so the talk is somewhat code-heavy
  - It is not however a regurgitation of the RooFit Doxygen: Google the classes to find out their syntax.
  - I will assume you are all familiar with the concepts of fitting: This talk is practical, not theoretical
  - The talk is by no means a complete introduction, RooFit is very extensive!
  - It should however give you a feel for RooFit and a starting point.
- Feel free to stop me if I've failed to explain something clearly







C. Fitzpatrick


February 16, 2015

# Why use RooFit?

- There are a number of features that make RooFit worth your while
  - Plenty of pre-written standard PDF components (gaussians, exponentials, crystal balls etc...)
  - You can build your PDFs up from these through addition, convolution, etc..,
  - When your PDF isn't easily factored into components it's easy to write a custom PDF
- Once you have a PDF written, toy studies are a walk in the park
  - RooFit handles the toy generation for you, with many features to make the process easier
  - You can generate with one set of parameters and fit to another set easily
  - Sensitivity and robustness studies are a few lines of code...
- Fitting to real data is just as easy
- RooFit handles blinding of parameters transparently



C. Fitzpatrick

February 16, 2015

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

# So how do you use RooFit?

- It comes bundled with ROOT, so any ROOT install will have RooFit
- RooFit is just an additional set of libraries with more functionality
- You can write root macros or compile binaries as you would normally
- Just make sure that in your macros you include the line:

```
using namespace RooFit;
```

- and if you compile binaries, make sure to include the necessary headers

```
#include "RooGlobalFunc.h"
```



C. Fitzpatrick

February 16, 2015

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

# The Basics

- When fitting, we usually deal with three basic object types:
  - **Data**: unbinned datasets, or binned histograms
  - **Variables**: These can be:
    - **Fit parameters**: eg; mean
    - **Observables** (dimensions) in phase space: eg; propertime
    - **Derived** Variables: eg; the result of a formula containing other variables
  - **PDFs**: Define the shape you're trying to fit, can be simple (a single Gaussian) or n-dimensional functions dependent on tagging, etc.
- The next few slides will show you how to construct and manipulate these objects in RooFit to perform a fit



C. Fitzpatrick

February 16, 2015

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

# RooRealVars

- A `RooRealVar` is the most basic non-derived real-valued object. It has a name, title, value, range and units
- A RooRealVar can be a fit parameter or can define the range over a given dimension in phase space to fit to.

```
RooRealVar x("x","x has range only",-10,10);
RooRealVar y("y","a constant called y",4.0);
RooRealVar z("z","range, initial value and units",4.0,-10,10,"miles");
```

- In the above example I've instantiated 3 RooRealVar objects:
  - `x` has been given only a range. This might be an ntuple column
  - `y` is a **constant**. It will never have a value other than 4.0
  - `z` has a unit, a range and an initial value. If this were used in a fit it will start at 4.0 but float to the fitted value.
- Sometimes it's useful to collect Variables together in a `RooArgSet` or `RooArgList` for passing to PDFs:

```
RooArgList Vars(x,y,z);
```

- A `RooArgSet` is an unordered collection, `RooArgList` is ordered



C. Fitzpatrick

February 16, 2015

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

# RooFormulaVars

- Derived variables in RooFit are called RooFormulaVars
- Almost anywhere you can use a RooRealVar a RooFormulaVar can be used instead
- Any floated RooRealVar input to the RooFormulaVar will then have the correct error and fitted value
- This is really useful for reparameterising fits to eg: reduce the correlations between parameters

```
// RooRealVars
RooRealVar a("a","a",1,0,5);
RooRealVar b("b","b",5,-10,20);

// We make a derived var that is the difference between a,b:
RooFormulaVar delta("delta","delta","a-b",RooArgList(a,b));

// In this instance we don't explicitly name the vars
// in the formula. Instead the vars are indexed @0...@n
// based on the order of the RooArgList
RooFormulaVar mean("mean","mean","(@0+@1)/2,0",RooArgList(a,b));

// We can now use "delta" and "mean" in our fit
// The fit will evaluate these so our result will be in terms of a,b
```

- The second form is generally better to use because if you rename `a,b` you would need to rewrite the first formula



C. Fitzpatrick

# RooPDFs

- There are a number of ways to make PDFs in RooFit:
  - Using any of the pre-written PDFs of which there are many (RooGaussian, RooBreitWigner, RooExponential, RooCBShape, RooBDecay, etc)
  - Writing your own custom PDF in RooFit is especially easy as you don't need to normalise it.
    - RooFit will do this for you numerically...
    - ...but if you can provide an analytic integral then it speeds things up
  - The third option is to mix and match. RooFit lets you add, convolve, take products of PDFs to build your models in a reasonably intuitive manner.
  - You can think of the process of building a PDF in RooFit as a series of modular steps:
    - What **should** your signal look like? Build a PDF for it...
    - What does your signal **really** look like? Include a resolution/acceptance on your PDF
    - What does your background look like? Build a PDF for it...
  - Then you can finally add the signal/background models together and try to fit!



C. Fitzpatrick

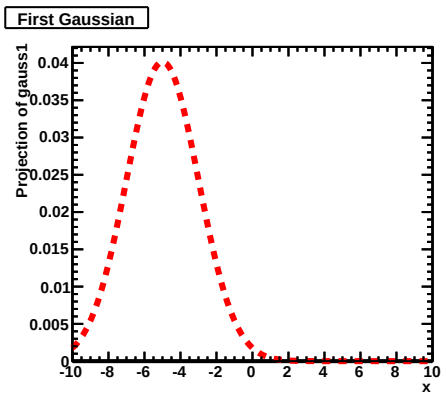
February 16, 2015

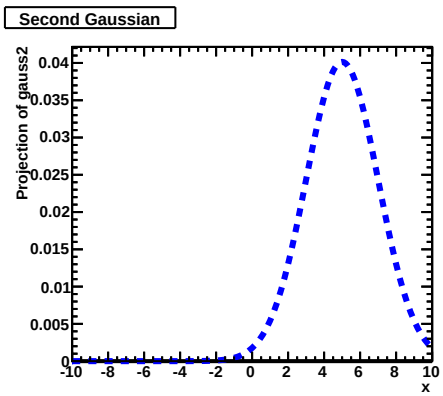
ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

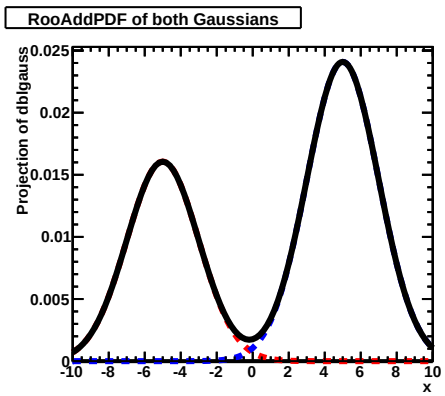
# RooAddPDFs

- When you want to **add** your signal/background models or make a PDF in one observable from components you use a `RooAddPDF`
- The PDFs added must have the same dimension

```
// The observable:
RooRealVar x("x","x",-10,10);

// A simple Gaussian PDF:
RooRealVar mean1("mean1","mean1",-5);
RooRealVar width1("width1","width1",2);
RooAbsPdf* gauss1 = new RooGaussian("gauss1","gauss1",x,mean1,width1);

// Another Gaussian with a different mean but in the same observable:
RooRealVar mean2("mean2","mean2",5);
RooRealVar width2("width2","width2",2);
RooAbsPdf* gauss2 = new RooGaussian("gauss2","gauss2",x,mean2,width2);

// We add these PDFs to make a new one as below. The frac variable
// specifies that 0.4/1.0 of the PDF area will be contained in the
// first gaussian
RooRealVar frac("frac","frac",0.4);
RooAbsPdf* dblgauss = new RooAddPdf("dblgauss","dblgauss",
                        RooArgList(*gauss1,*gauss2),frac);
```







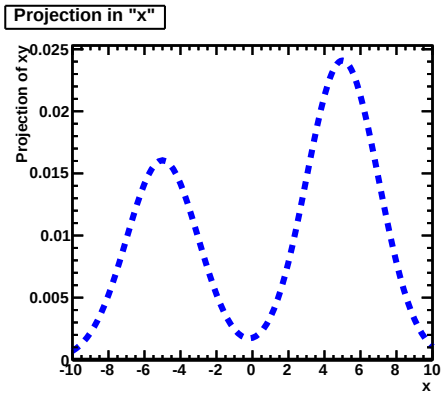
C. Fitzpatrick

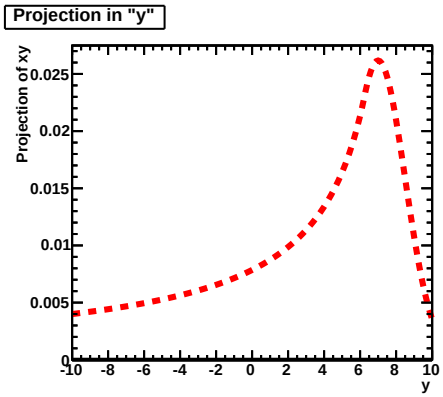

February 16, 2015

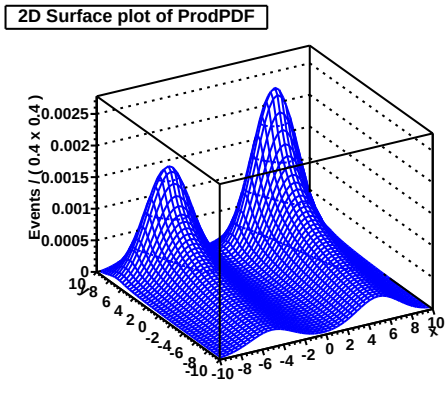
# RooProdPDFs

- When you want to fit across more than one dimension you need a `RooProdPdf`:
- Here I've reused the AddPDF in the "x" observable, and included a new PDF in the "y" observable:

```
// Now we create a new observable, y, and a crystal ball PDF
RooRealVar y("y","y",-10,10), m0("m0","m0",7), n("n","n",0.9);
RooRealVar sigma("sigma","sigma",1.5), alpha("alpha","alpha",0.5);

RooAbsPdf *cball = new RooCBShape("cball","cball",y,m0,sigma,alpha,n);

// We take the RooAddPDF we just made and the crystal ball together
// To make a new PDF in 2 dimensions:
RooAbsPdf *xy = new RooProdPdf("xy","xy",*dblgauss,*cball);
```




- This example assumes no correlation between observables x,y
- For the case in which x,y **are** correlated see the `rf305_condcorrprod.C` tutorial




C. Fitzpatrick


February 16, 2015

# Rolling your own: RooGenericPdfs

- There are two ways to hand-code your own PDF if RooFit's own ones aren't up to scratch
  - The quick and dirty option is to write a `RooGenericPdf` in-line
  - For more complicated PDFs it is better to write your own shared library
- RooGenericPDFs Take a RooFormulaVar style formula string and a RooArgList of variables:

```
// Define a mean, width and observable:
RooRealVar mu("mu","mu",5,-10,10), sigma("sigma","sigma",2,0,5);
RooRealVar obs("obs","obs",-10,10);
RooArgList gaussList(obs,mu,sigma);

// Make a simple gaussian RooGenericPDF:
RooAbsPdf *myGaussian = new RooGenericPdf("myGaussian","myGaussian",
                "exp((-0.5*(@0-@1)*(@0-@1))/(@2*@2))",gaussList);
```

- You don't need to (and shouldn't) normalise your PDF, as RooFit will do the integration/normalisation for you
- In the case of a RooGenericPdf this is always done numerically, so compilcated PDF models will take time to generate/fit



C. Fitzpatrick

February 16, 2015

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

# Rolling your own: Custom PDF libraries

- A more powerful and easier to debug method is to write your own `RooAbsPdf`
- This allows you to optionally specify analytic integrals, generator speedups and use C++ functions in-situ
- You write a header and `.cxx` file which you then pass through root to make a shared library
- You can use the `RooClassFactory` to do most of the work for you
- I'll not go into further detail here as this is all well explained in the roofit tutorial `rf104_classfactory.C`
- I also recommend you take a look at the source code of the pre-written RooFit PDFs as they are formatted in the way your own PDFs should be



C. Fitzpatrick

February 16, 2015

# RooDataSets

- A RooDataSet is RooFit's data storage container for unbinned data
- It exists because ROOT NTuples can't be memory resident.
- To get from NTuple to RooDataSet you should only load the columns you'll be fitting to as it saves memory:

```
// Open a DaVinci DecayTreeTuple:
TFile *rootFile = new TFile("DVNtuple.root");
TTree *ntuple = (TTree*)rootFile->Get("DecayTreeTuple/DecayTree");

// Create RooRealVars with the same names as columns in the ntuple
RooRealVar phi_mass("phi_1020_MM","M(K^{+}K^{-})",990.0,1070.0);
RooRealVar bs_mass("B_s0_MM","M(J/#psi #phi)",5300,5400);

// Make a RooArgList of these Vars:
RooArgList phaseSpace(phi_mass,bs_mass);

// And now turn the ntuple into a RooDataSet
// containing only the Vars in phaseSpace
RooDataSet *data =
        new RooDataSet("data","data from Ntuple",ntuple,phaseSpace);
```

- A RooDataHist is the binned equivalent. It is often much faster to fit to these, but the usual caveats regarding binned fits apply
- To turn a TH1, TH2 or TH3 into a RooDataHist:

```
RooDataHist datah("datah","binned data from TH3",RooArgSet(x,y,z),myTH3);
```

- The same instantiation will turn a RooDataSet into a RooDataHist, but you need to specify the binning first:

```
x.setBins(10); y.setBins(20); z.setBins(10);
```



C. Fitzpatrick

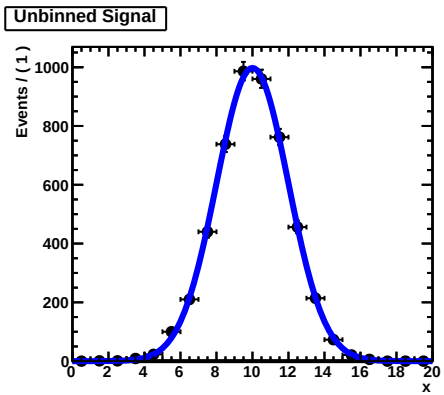
February 16, 2015

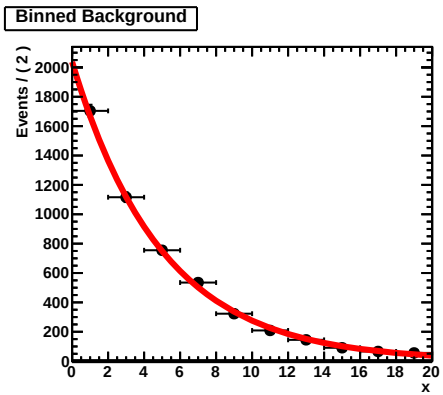
ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

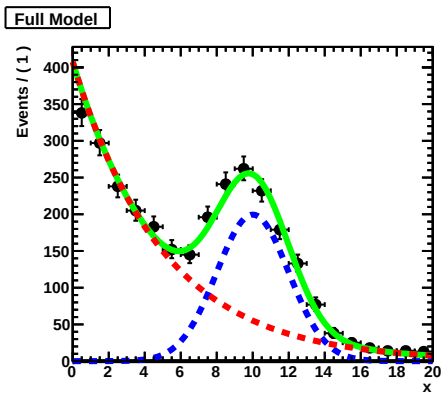
# Generating data

- Once you have a PDF written, it is trivially easy to generate data from it
- RooFit chooses amongst several generation methods automatically to speed the process up

```
RooAbsPdf* signal = new RooGaussian("signal","signal",x,mean,sigma);
RooAbsPdf* bkg = new RooExponential("bkg","bkg",x,coeff);

// Generate 5000 signal events, verbose flag set to true:
RooDataSet* signalData = signal->generate(RooArgSet(x),5000,true);

x.setBins(10);  // Set binning for binned data
// Generate a binned datahist equivalent to 5000 bkg events:
RooDataHist* bkgData = bkg->generateBinned(RooArgSet(x),5000,false,false);

// Here I use the extended likelihood formalism:
// I explicitly state signal and background yield terms:
RooRealVar Nsig("Nsig","Nsig",1000), Nbkg("Nbkg","Nbkg",2000);
RooAbsPdf* model = new RooAddPdf("model","model",
                RooArgList(*signal,*bkg),RooArgList(Nsig,Nbkg));

// Generate the full signal/background distribution in one dataset
// using the yields specified in the extended model:
RooDataSet* modelData = model->generate(RooArgSet(x),0,true);
```







C. Fitzpatrick


February 16, 2015

# Fitting data

- There are many ways to fit in RooFit:
  - RooFit's default method is a -ve log likelihood minimisation using minuit
  - If the data is unbinned, the fit will be unbinned. For RooDataHists it will be a binned fit.
  - You can optionally do extended fits to estimate yields, binned $\chi^2$, use minos for 2-sided errors etc..
- All of this is performed and controlled using the `RooAbsPDF::fitTo()` function which returns a `RooFitResult*` object:

```
// Fit our model to the dataset we just generated, saving the fit result:
RooFitResult *result = model->fitTo(*modelData,Save(kTRUE));

// Check the MINUIT covariance quality:
if( result->covQual() == 3){
        // If it's good, print a nice LaTeX table of only the floated vars
        result->floatParsFinal().printLatex(Format("NEAU"));
}else{
        // If it's bad, warn us
        cout << "ERROR: covariance quality not 3," << endl;
        cout << "fit result may be inaccurate!" << endl;
}
```

- After fitting, any floating parameters are set to their fitted value.
- The RooFitResult stores the covariance matrix, initial and final values of fit parameters, minuit fit status, etc.
- This is very handy: You can use the updated parameters in a later fit or reset them, and you can compare fit results easily

- RooFit has built-in LaTeX table formatting, so the result is easily printed and copied into your publications

| | |
|---|---|
| Nbkg | $2053 \pm 67$ |
| Nsig | $947 \pm 59$ |
| coeff | $-0.19652 \pm 0.0084$ |
| mu | $9.747 \pm 0.098$ |
| sigma | $1.93 \pm 0.11$ |




C. Fitzpatrick

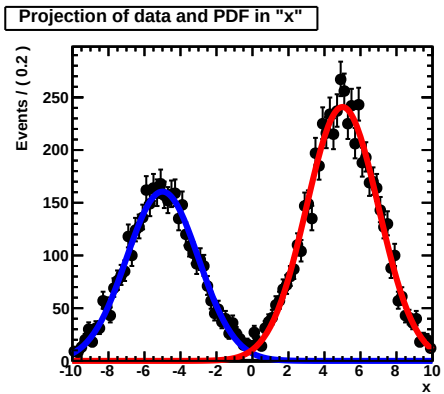

February 16, 2015

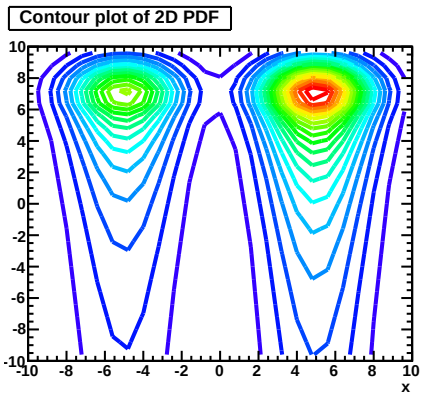

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

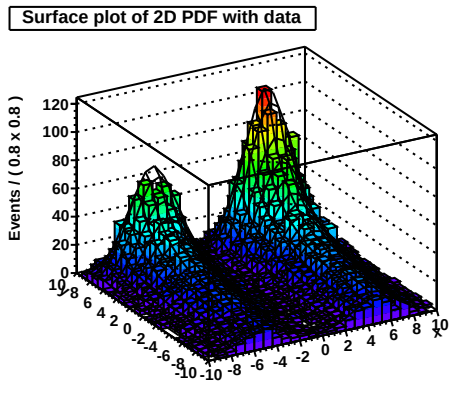
# Plotting your results

- 1D projections of data and PDFs are implemented in a `RooPlot`
- To visualise in 2D/3D RooFit returns a ROOT `TH1` object

```
// Generate data from the 2D model shown earlier
RooDataSet *xydata = xy->generate(RooArgSet(x,y),10000,true);

// Make a RooPlot with the x observable as the axis and plot the data:
RooPlot *xproj = x.frame(Title("Projection of data and PDF in \"x\""));
xydata->plotOn(xproj);

// plot the 2 gaussian components of our model
xy->plotOn(xproj,Components(*gauss1),LineColor(kBlue));
xy->plotOn(xproj,Components(*gauss2),LineColor(kRed));
xproj->Draw();

// Get the TH1 of the model and plot it as a contour plot in 2D:
TH1* xymodelhist =
        xy->createHistogram("xymodel",x,Binning(25),YVar(y,Binning(25)));
xymodelhist->Draw("cont1");

// Get the data as a TH1, plot both data + model as lego and surface:
TH1* xydatahist =
        xydata->createHistogram("xydata",x,Binning(25),YVar(y,Binning(25)));
xydatahist->Draw("lego2 0"); xymodelhist->Draw("surf same");
```





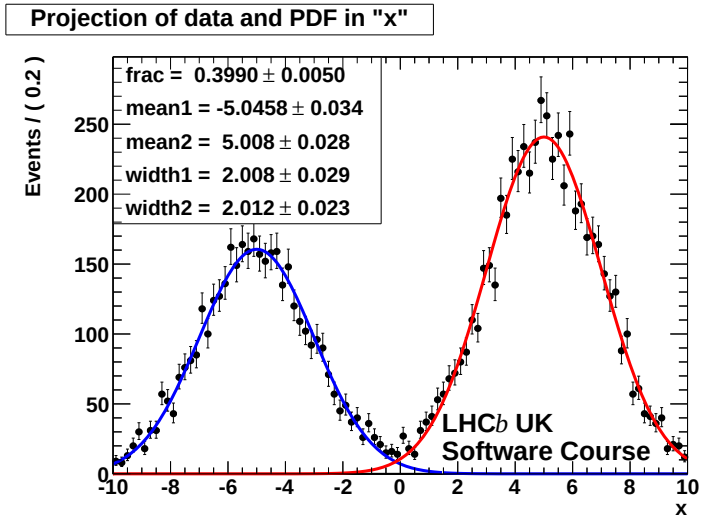


C. Fitzpatrick


February 16, 2015

# Adding results and text to plots

- You can add the result of your fit to your RooPlots with the `RooAbsPdf::paramOn()` command
- This takes many optional arguments, but the default is to display only parameters that were floated in the fit
- Adding labels is identical to that in a ROOT TCanvas




```
// use paramOn to display the fit result of floated parameters
// on the x projection plot:
xy->paramOn(xproj,AutoPrecision(1),Layout(0.15,0.5,0.90));

//Add a label to the plot:
TPaveText * label = new TPaveText(0.58, 0.15, 0.58, 0.25,"BRNDC");
label->SetBorderSize(0); label->SetFillColor(0);
label->SetTextAlign(11); label->SetTextSize(0.05);

TText * labeltext = 0;
labeltext = label->AddText("LHC#font[12]{b} UK");
labeltext = label->AddText("Software Course");

xproj->addObject(label); xproj->Draw();
```




C. Fitzpatrick

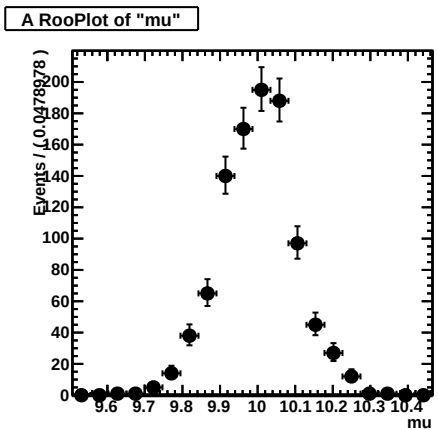

February 16, 2015

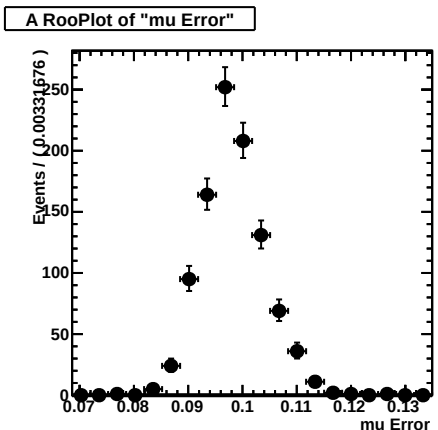

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

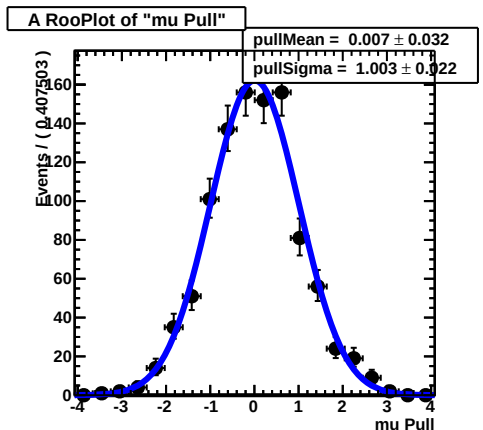
# Toy studies

- This is one of the more elegant aspects of RooFit
- The `RooMCStudy()` class handles all the fitting and generating for you
- Here I present the simplest case
- I recommend that you take a look at roofit tutorials `rf801-rf804` to see more examples

```
// Take our simple gauss + expo model from the dataset example
// earlier and perform a toy study with it:
RooMCStudy *study = new RooMCStudy(*model,RooArgList(x),
        Binned(kTRUE),  //Do a binned fit to speed things up
        FitOptions(Save(kTRUE)), //Save the RooFitResults
        Silence()); //Verbosity suppression

// generate 1000 toy datasets and fit to them
study->generateAndFit(1000);

// Plot the fitted value of the mean parameter for all 1000 toys:
RooPlot *value = study->plotParam(mean,Bins(20));
// Plot the error on the mean as well:
RooPlot *error = study->plotError(mean,Bins(20));
//Calculate and plot the pull, fit a gaussian to it:
RooPlot *pull = study->plotPull(mean,Bins(20),FitGauss(kTRUE));
```







C. Fitzpatrick

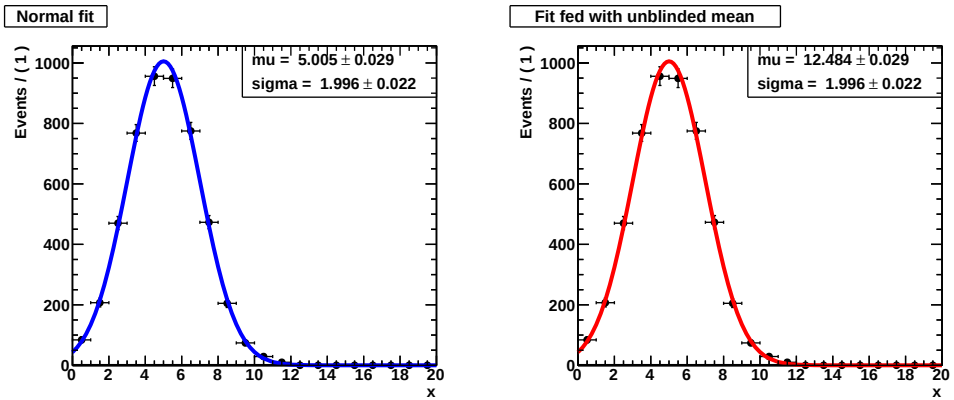

February 16, 2015


ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

# Blinding

- Blinding is implemented by replacing a `RooRealVar` with a `RooAbsHiddenReal`
  - The RooRealVar becomes blinded, with the unblinded value stored in the RooAbsHiddenReal
  - The blind parameter will have an error equivalent to the unblinded one

```
//A standard gaussian pdf:
RooAbsPdf* genpdf = new RooGaussian("genpdf","genpdf",x,mean,sigma);
//We generate a dataset from this and fit to it as usual
RooDataSet* data = genpdf->generate(RooArgSet(x),5000,true);
RooFitResult *result = genpdf->fitTo(*data,Save(kTRUE));

// Now we make an unblinding transformation:
//ub_mean will hold the unblinded (actual) value of the mean
TString blindstring = "post␣tenebras␣lux";
RooAbsHiddenReal *ub_mean =
        new RooUnblindUniform("ub_mu","ub_mu",blindstring,10,mean);
// We fit a new PDF using the unblind var to hold the true mean:
RooAbsPdf* fitpdf = new RooGaussian("genpdf","genpdf",x,*ub_mean,sigma);
RooFitResult *blindresult = fitpdf->fitTo(*data,Save(kTRUE));

cout << "Unblinded␣value␣is:␣" << ub_mean->getHiddenVal() << endl;
```







C. Fitzpatrick


February 16, 2015

# Documentation & help

- The RooFit Doxygen is incorporated into the standard ROOT Doxygen pages
  http://root.cern.ch/root/html528/RooFit.html
  - It isn't bad if you know what class you're using but don't know the syntax
  - Usually, googling the class you're interested in will result in the doxygen page you want.
- The Roofit homepage is
  http://roofit.sourceforge.net/
  - This hasn't been updated since 2006 but the code has, so beware!
  - It is home to the RooFit User's Guide and some slideshows that can be conceptually helpful, but don't expect everything to work!
- The Official RooFit tutorials are quite good and much broader in scope than I have had time for here. You can find them in `$ROOTSYS/tutorials/roofit`
- Lastly, if you're really stuck, the root talk forums "Stat and Math Tools" subforum is a good place to ask questions
  http://root.cern.ch/phpBB3/viewforum.php?f=15
  - You'll often get an answer from a RooFit developer if you ask in here
    - Make sure you provide a short code example they can run or modify,
    - tell them what version of RooFit you're using, etc.
  - Search the forum first to see if your question has already been answered.
- For help with the theory of fitting, probablility and statistics, these slides by R. Barlow are very nice: here and here



C. Fitzpatrick

February 16, 2015

# Concluding Remarks

- I hope that the contents of this talk have given you a feel for how RooFit works
- It is only the tip of the iceberg however, RooFit is incredibly extensive
- In addition to the topics covered here I'd like to point out a few of the prepackaged RooFit tutorials on more advanced subjects:
  - `rf208_convolution.C`:
    RooFit uses FFTW3 to make convolution easy
  - `rf703_effpdfprod.C`:
    Including an acceptance function without losing generator efficiency
  - `f501_simultaneouspdf.C`:
    Constructing a simultaneous fit to several datasets
  - `rf505_asciicfg.C`:
    Control your fits using a simple ascii config file



C. Fitzpatrick

February 16, 2015

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

# The tutorial

- The tutorial is written in several steps
  - The idea is that you uncomment and complete the code as you go
  - The contents of this talk should be sufficient to complete the tutorial...
  - but be prepared to look up the RooFit Doxygen in order to understand syntax, etc.
- You can find the tutorial here:
  `http://void.printf.net/~conor/epfl_roofit_tutorial.tar.gz`
- copy this to your workspace, and extract it:
  ```
  tar -xvf epfl_roofit_tutorial.tar.gz
  cd epfl_roofit_tutorial
  ```
- Inside you'll find `dataset.root` and `roofit_tutorial.C`
- open `roofit_tutorial.C` in your favorite editor, and read the comments
- When you want to run the example:
  `root -x roofit_tutorial.C`



C. Fitzpatrick

February 16, 2015

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE